IN THE CLAIMS

1. (Original) A computing system for creating an extensible N-tiered software application, comprising:

a. at least one processing unit;

b. at least one memory store operatively connected to the processing unit;

c. extensible N-tiered creation software, executable within the at least one processing unit, comprising a plurality of predetermined extensible N-tier architecture rules; and

d. at least one extensible tier, capable of residing in the memory store, further comprising:

i. a framework that specifies a basic design structure for software components categorized as belonging to the extensible tier, the framework further comprising base software components and a set of standard interfaces for any software component categorized as belonging to the tier; and

ii. a logically grouped set of a predetermined number of executable software components compliant with the tier framework, each software component further capable of communicating with at least one other software component.

2. (Original) The computing system of claim 1 further comprising a communications network, operatively in communication with the processing unit, the communications network selected from the group of networks consisting of local internal

networks, local area networks, asynchronous networks, synchronous networks, and wide area networks.

3. (Original) The computing system of claim 1 wherein communication between the software components comprises asynchronous communications.

4. (Original) The computing system of claim 1 further comprising an inventory of software components.

5. (Original) The computing system of claim 1 wherein the at least one extensible tier is a set of extensible tiers, further comprising a set of logical connections comprising sequencing and messaging information between a first one of the extensible tiers and at least one other tier of the extensible set of tiers, whereby each tier in the extensible set of tiers is capable of communicating with any other tier through one or more tier interfaces.

6. (Currently amended) The computing system of claim 5 wherein the set of extensible tiers comprises a base tier comprising a base tier framework, the base tier framework comprising:

a. at least one collection interface for collecting software components, including software components which are normally aggregated into other software components;

b. at least one connection interface for connecting software components as sources or sinks of information;

c. at least one messaging interface comprising message-based behavior; and

d. at least one control interface for controlling devices.

7. (Original) The computing system of claim 5 wherein the set of extensible tiers comprises a business rules tier, a processing tier, a data tier, a messaging tier, a business objects tier, a visual tier, a base tier, a real-time device tier, an interceptor tier, and an application tier.

8. (Currently amended) The computing system of claim 7 wherein the processing tier further comprises a framework comprising:

a. an interface capable of handling processing components associated with the processing tier;

b. an aggregation interface to aggregate a predetermined number of software component attributes;

c. a query interface comprising a query modification interface; and

d. a predetermined number of windowed input/output parameters to satisfy processing tier requirements;

e. whereby a processing tier software component can manage process flows and monitor and/or optimize flow of data through a model.

9. (Original) The computer system of claim 8 further comprising a tracking interface to track processing model filter requirements.

10. (Original) The computer system of claim 9 wherein the processing components comprise filters, synchronization components, sources, sinks, and graphical components.

11. (Currently amended) The computer system of claim 7 wherein the data tier comprises a data tier framework and provides data persistence services for a predetermined set of software components and access to data, the data tier framework further comprising:

a. a data modification interface whereby data may be written to and read from a data source; and

b. a data access interface whereby access may be provided to specific types of data.

12. (Currently amended) The computer system of claim 7 wherein the messaging tier further comprises a messaging tier framework and messaging software components to convey information from a source of messages to a recipient of messages, the messaging tier framework further comprising:

a. a message generation interface;

b. a message queuing interface;

c. a message routing interface;

d. a message text management interface;

e. a message routing interface whereby one or more software components that will receive a message may be specified; and

f. a message queue interface whereby information about a specific queue may be specified;

g. wherein the messaging software components control message queuing and notification, and support message generation for different types of messages.

13. (Currently amended) The method computer system of claim 12 wherein the messaging tier interface supports asynchronous messaging.

14. (Currently amended) The computer system of claim 7 wherein the business components tier further comprises:

a. at least one business software component, the business software component comprising a general purpose data container providing storage for and access to information, the business software component further encapsulating attributes comprising data and behavior for a business entity; and

b. a business software component framework, comprising:

i. a model comprising a collection of related business software components to reflect a real-world business entity; and

ii. a binary large object to allow storage of large amounts of data within the business software component.

15. (Currently amended) The computer system of claim 14 wherein at least one business software component has an attribute that is other business software component.

16. (Currently amended) The computer system of claim 14 wherein the at least one business software component is a composition defining a static model whereby the relationship

between the business software component and an attribute component is not an association enforced by business rules.

17. (Currently amended) The computer system of claim 14 wherein the model further comprises a set of business software components model interfaces that the business software components aggregate to achieve specific functionality.

18. (Currently amended) The computer system of claim 17 wherein the business software components model interfaces comprise:

a. an association interface whereby business software components may be associated with other software components;

b. a relationship interface whereby hierarchical relationships with other software components may be established and maintained;

c. a data dump interface whereby contents of a software component may be selectively retrieved; and

d. a name interface whereby persistable attribute names may be retrieved.

19. (Currently amended) The computer system of claim 7 further comprising a visual tier to provide display of and user interaction with information, the visual tier using a model, view, and controller design pattern comprising:

a. a modeler comprising data and computational logic to handle user interaction, the modeler further comprising an event handler, a connection source, and a connection sink;

b. a viewer comprising data and computational logic to present at least a portion of a model perceptively to a requestor, the view component further comprising an event handler and a connection source; and

c. a controller comprising data and computational logic to handle requestor interaction, the requestor interaction further comprising actions from an input device, the controller component further comprising an event handler, a connection source, and a connection sink;

d. wherein the modeler, the viewer, and the controller may utilize a messaging tier whereby each of the modeler, the viewer, and the controller can have one or more message handlers attached to provide additional behavior, the modeler and the controller having different message handlers attached to effect differing behavior.

20. (Currently amended) The computer system of claim 19 wherein the visual tier further comprises:

a. a read-only mode, whereby the viewer can be specified to allow read-only access to the system; and

b. an editing mode, whereby users may edit models as well as view them;

c. wherein an N-tiered application may selectively comprise either the read-only mode, the editing mode, or both on a user by user basis.

21. (Original) The system of claim 7 wherein the real-time device tier further comprises:

a. a communication interface; and

b. an event-handling interface;

c. whereby a real-time device can communicate with connected software components to support communication with and event handling for a real-time device.

22. (Currently amended) A method for ~~generating~~ a ~~software application in a~~ computing system for creating an extensible N-tiered software application for a system including comprising at least one processing unit; and at least one memory store operatively connected to the processing unit; the method comprising:

utilizing extensible N-tiered creation software, comprising a predetermined set of software component rules, tier rules, and application assembly rules, the N-tiered software being executable within the at least one processing unit;

employing an inventory of executable software components, each software component further comprising a given structure and an external interface and further capable of communicating with at least one other software component; and a predetermined set of initial extensible tiers capable of residing in the memory store, each tier of the predetermined set of extensible tiers having a given structure, the set of extensible tiers further comprising a logically grouped set of a predetermined number of the executable software components, ~~the method comprising:~~

~~a. determining a set of application requirements;~~

~~b. for each of the set of application requirements, reviewing the inventory of software components for software components that match at least one of the set of application requirements;~~

c. for each application requirement in the set of application requirements for which a software component match does not exist in the software component inventory, obtaining a software component that does match that application requirement;

d. defining a set of tiers to logically model the application requirements;

e. selecting tiers from the predetermined set of tiers to satisfy the defined set of tiers;

f. for tiers not within the predetermined set of tiers needed to satisfy the defined set of tiers, creating new tiers;

g. associating each of the matching software components with at least one tier of the defined set of tiers according to a framework associated with that tier; and

h. creating a software application by assembling the predetermined set of tiers according to the application assembly rules;

i. whereby the software application satisfies the set of application requirements.

23

23. (Original) The method of claim 22 wherein obtaining a software component that does match that application requirement further comprises selectively modifying an existing software component or procuring and selectively modifying a new software component from an independent source of software components to comply with a tier's framework requirements.

24. (Original) The method of claim 22 further comprising:

a. examining the obtained software components for incorporation into the software component inventory according to predetermined incorporation criteria; and

b. storing the obtained software component in the software component inventory if it meets the incorporation criteria.

25. (Original) The method of claim 22, wherein at least one of the software components is a business software component, further comprising automatically retrieving composite software components along with an associated business software component when the associated business software component is retrieved from a persistent store.

26. (Currently amended) The methodsystem of claim 22 wherein composite components are not created when a new business software component is created but instead where it is the responsibility of the creator to create a composite component and set it into the composing component.

27. (Original) The method of claim 22 wherein software components and tiers are combined at run-time to form new, unique applications on-the-fly.

28. (Original) The method of claim 22 further comprising:

a. defining a testing tier;

b. testing a final model using the testing tier; and

c. correcting errors within the executable software components within the final model.

29. (Original) The method of claim 22 wherein associating is accomplished using a graphical user interface.

30. (Original) The method of claim 22 wherein creating new tiers further comprises:

a. examining a requirement;

b. determining if a current framework is adaptable to accommodate the requirement;

c. using the current framework if it is adaptable to accommodate the requirement;

d. defining a new framework to accommodate the requirement is no current framework is adaptable or otherwise accommodates the requirement; and

e. creating a new tier with the new framework.

31. (Original) The method of claim 22 further comprising removing a tier, the step comprising:

a. examining current requirements;

b. for each current tier, determining if at least one other tier satisfies the requirements;

c. if so, combining those tiers;

d. for each remaining tier, determining if the tier is no longer needed to satisfy at least one requirement; and

e. if so, remove the no longer needed tier.

32. (Original) The method of claim 22 further comprising defining an initial set of tiers.

33. (Original) The method of claim 32, wherein the initial set of tiers comprise a business rules tier, a processing tier, a data tier, a messaging tier, a plotting tier, a business

objects tier, a visualization tier, a base tier, a real time device tier, an interceptor tier, and an application tier.

34. (Original) The method of claim 33 wherein defining a business rules tier framework further comprises:

a. defining a set of business rules;

b. defining a predetermined set of properties and methods which can be used to determine whether a first software component violates the business rules by being associated with a second software component;

c. defining a predetermined set of properties and methods which can be used to determine a type of each associated child software component;

d. defining a predetermined set of properties and methods which can be used to determine a type of each associated parent software component; and

e. defining a predetermined set of properties and methods which can be used to determine whether a first software component having a specific type violates the business rules by being associated with a second software component having a specific type.

35. (Original) The method of claim 33 further comprising allowing building of business software components as general purpose, reusable data containers, whereby data and behaviors in a business software component are hidden from other software components, other components are kept from having to know the business software component's internal structure

and implementation details, and data integrity checks are allowed to occur at the business software component level.

36. (Original) The method of claim 35 further comprising collecting business software components into a heterogeneous collection model that represents a real-world business entity.

37. (Original) The method of claim 35 wherein collecting business software components uses a generalized collection interface to collect other business software components.

38. (Original) The method of claim 33 further comprising specifying a framework for the real-time device tier to support communication with and event handling for at least one real-time device, the framework comprising properties and methods that support a predetermined communication interface and a predetermined event-handling interface whereby a real-time device can communicate with connected software components.

39. (Original) The method of claim 33 wherein the interceptor tier comprises properties and methods to allow interception and control of messages passed between software components or calls to a software component's interface, whereby an intercepted message or interface call may be validated, interrogated, and acted upon by callbacks registered with the interceptor tier before the intercepted message or interface call is transmitted to a target software component, allowing validation and control of the disposition of the message or call to the

software component's interface to occur without modifying a source or the target software component.

40. (Original) The method of claim 39 further comprising:

a. specifying a method for registering a callback with the interceptor;

b. specifying a method for canceling registration of a callback; and

c. specifying a method for creating an instance of a software component.

41. (Currently amended) The method of claim 33 further comprising:

a. specifying a method within the application tier for creating applications that use asynchronous behavior;

b. specifying how software components are created and registered; and

c. _____specifying how service components are connected;

d. whereby dependencies and communication links are established before an application begins responding to events.

42. (Original) The method of claim 33 further comprising specifying a wizard tier comprising one or more wizards developed for frameworks to insure that proper framework interfaces are implemented for a software component associated with a tier, whereby the wizard tier may be used during a development process.

43. (Original) The method of claim 33 further comprising:

a. specifying a set of rules and required activities for the testing tier; and

b. using the rules and required activities to define acceptable tests of a software component;

c. wherein the rules and activities comprise at least one test harness to run a test script and store test results, whereby the test harness inspects a software component associated with [a framework to insure that the software component has implemented all the required interfaces for that framework and that the interfaces function properly.

44. (Original) The method of claim 43 wherein the rules and required activities are defined by and derived from the requirements for developing a software component.

45. (Original) The method of claim 33 further comprising providing a template tier, the template tier comprising templates comprising predetermined software language implementations of persistence, collections, and iterators for software components within the template tier, whereby the templates are used in component implementation to facilitate the implementation of predetermined functionality and to reduce the maintenance effort for extending the functionality of components.

46. (Original) A method for generating an application comprising software components, each software component having a given structure, comprising:

a. determining an application's system requirements;

b. with the system requirements, creating one or more tiers from an initial set of tiers to create a model design, each tier being responsible for providing a discrete set of

application programmatic responsibilities, until all the tiers collectively satisfy the application's system requirements;

c. for each tier, creating a framework, comprising:

i. defining an architect context for software components to be associated with that tier, comprising specifying a basic design structure, including base components, of that tier;

ii. defining a logical grouping of executable software components to be associated with that tier to implement the tier; and

iii. specifying a set of standard interfaces for any software component categorized as belonging to that tier;

d. for each framework:

i. checking each required software component for existence in an inventory of software component components;

ii. selecting for use within a tier software components within each framework that are also present in the inventory that satisfy the framework; and

iii. for each required software component not present in the inventory, obtaining a software component to satisfy the requirement;

e. for each tier:

i. assembling all software component components associated with the tier into that tier; and

ii. defining the sequencing and data relationships between that tier and each other tier with which that tier needs to be sequenced or exchange data; and

f. packaging all the tiers that collectively satisfy the application's system requirements into an invocable application.

47. (Original) The method of claim 46 further comprising:

a. defining a testing tier;

b. testing the invocable application using the testing tier; and

c. correcting errors within the invocable application.

48. (Original) The method of claim 46 wherein defining the sequencing and data relationships between that tier and each other tier with which that tier needs to be sequenced or exchange data are accomplished via a graphical user interface.

49. (Original) A computer program embodied within a computer-readable medium created using the method of claim 22.

50. (Original) A computer program embodied within a computer-readable medium created using the method of claim 46.